The Study on Distributed Speech Recognition System

WeiQi Zhang, Liang He*, Yen-Lu Chow**, RongZhen Yang*, and YePing Su**

*Intel Architecture Development Lab

** Lernout & Hauspie Asia Pacific

Abstract

This paper focuses on the architecture design, implementation and optimisation of distributed speech recognition systems. Client-server DSR architecture is presented based on the analysis of alternative architectures and integrated prototype system is investigated in client, network and server parts. Finally system optimization is considered in transmission bandwidth, server workload control and client security aspects.

1.Introduction

With the rapid growth of speech and Internet technology, the DSR applications brought out many research and engineering efforts [1][2]. Among the numerous studies of speech enabled applications over the web, a practical fundamental architecture was described and analysed in [3]. The industry standard for DSR front-end processing in client has been investigated since 1995 and probably will be defined by the end of 2000 [4]. The W3C Voice Browsers Working Group was formed in Oct. 1998 to serve as a coordination body with existing industry groups working on the related specifications such as dialog management, extensions to existing Web standards, speech grammar formats and authoring guidelines [5]. Some demo systems and next generation products can be found in [6][7].

DSR involves many diversified technologies including ASR (automatic speech recognition), natural language understanding, network data transmission bandwidth and protocol, multithread, distributed computing and etc. So detailed and comprehensive studies on the Framework of DSR are necessary.

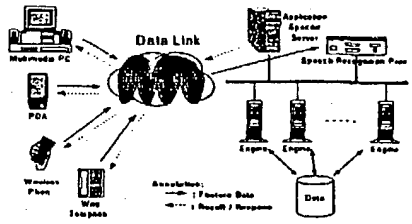

Fig1. The Principle Framework of DSR

The principal client-sever framework of DSR is represented in Fig 1. Various kinds of clients such as multimedia PCs, PDAs, wire and wireless telephones can act as speech-enabled client devices. With proper network bandwidth and protocols, the speec features and recognition results are transmitted between clients and servers via PSTN and Internet. In the server side, the multi-users speech recognition jobs are dispatched b Speech Recognition Proxy (SRP), the actual recognition is processed in engine servers, the processing of language understanding for relative response achievement and results transmission are done in application specific servers.

In this paper, the fundamental architecture of DSR is discussed in chapter 2. The novel mechanism of DSR realization, including client functional modules, networks QoS and congestion control and distributed recognition server, is presented in chapter 3. Some optimization suggestions are proposed in chapter 4.

2. Fundamental Architecture of DSR

Generally there are three alternative strategies in the design of DSR architectures. The first is server-only processing in which all processing are done at the server side and the speech signal is transmitted to the server either through the Internet by using speech coding or via a second channel like telephone [6]. The second is client-only processing in which most of speech processing is done at the client side and then the results are transmitted to the server. The third is client-server processing. In this model front-end processing is done at the client side, then the speech features are transmitted to the server and finall the processing of speech decoding and language understanding are performed at the server side [4][7].

Based on the considerations of speech transmission bandwidth and recognition rate, above three strategies are evaluated in [3]. As to the server-only processing model, pure voice transmission requires high-speed network bandwidth, low bandwidth connections will cause recognition performance degradation. The client-only processing limits type of clients that are powerful enough to perform the complicated speech recognition. Comparing with above two models, Client-server processing model has many benefits as follow

- MFCC is a common set of features extractor used b many state-of-the-art HMM based ASRs. It can be implemented in many types of clients and hopefully will be defined as standard for industry compatibility [4].
- Speech feature efficiently carries the information which speech recognition needs. Its data size is much smaller than pure voice.

The proposed fundamental DSR architecture is shown in Fig 2.

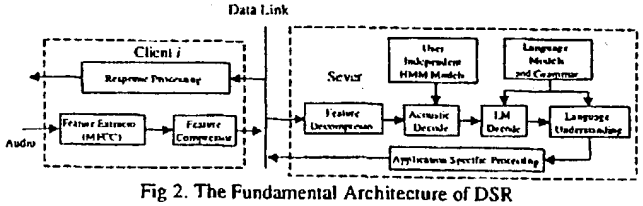

Fig 2. The Fundamental Architecture of DSR

In a representative case [10], the dimension of the features per frame is 13 and the frame rate is about 100 per second. In the following, we will discuss bandwidth requirements of speech feature transmission in different HMM methods.

- Discrete-Density HMM (DHMM). There are six codebooks for energy, cepstral coefficients, first-order and second-order difference of cepstral coefficients. The

number of codewords for cepstral coefficients codebook and their difference codebooks is 256. The number of codewords for three energy codebooks is 32. If this DHMM codebooks set is used, the required transmission bandwidth is (3 x 8 + 3 x 5) x 100 = 3.81 kbps. This bandwidth is much lower than speech coding transmission bandwidth but much higher recognition rate is achieved [3].

- Continuous-Density HMM (CHMM). CHMM uses acoustic parameter vector (MFCC) directly. The transmission bandwidth should be 13 x 32 x 100 = 40.625 kbps. It may not be acceptable by many current network settings. After the effective generalized Lloyd subvector quantization, the bandwidth can be reduced to 2.0 kbps without increasing error rate much [3].

3.The feasible realization mechanism of DSR

Building a practical DSR system can become a complex task because it should have the capability of multi-user support, flexibility, high performance, reliability, scalability, eas upgrade and low cost, and etc[12]. In this chapter, we integrate solutions across three sections: client, network and server to present the realization mechanism with above characteristics.

3.1Voice enabled Client

As the analysis in chapter 2, client will send the speech features vector to server. Taking the idea in paper [9], the client can be separated into five functional components: voice recorder, feature extractor, feature compressor, data transmitter and TTS (see fig3).

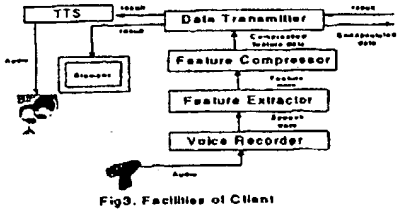

Fig3. Facilities of Client

When speech recognition application is enabled, voice recorder program is invoked to record and send audio samples to Feature Extractor. Then the feature extractor and compressor calculates speech feature as described in chapter 2. Data Transmitter is responsible for sending features vector and receiving responses. If the sound response is needed (such as telephones) TTS takes its place to translate text result into speech.

There are different implement ways to build above five functional components and they aim for different client scenarios. These technologies may be integrated according to specific applications and special user facilities

- Mobile agent or mobile code. MAs and MCs such as Java Applets are easy to implement, but they bear the disadvantage of slow execution speed. Without browser and enough memory, telephone clients have to use JVM-based MC with industry standard DSPs in the near future.
- Plug-in. Fast-running p latform-dependent plug-ins are configured as browser's installed capabilities and can be reused [8]. Only clients such as PCs and PDAs with browsers can support plug-ins. Comparing with MC's low-performance and CORB 's complexity, plug-ins are optimal for Unix clients.
- Common Object Model (Only on MS Windows). For most of Windows users, COM is the best choice because it provides high performance DSR service whether in or out of browsers.
- CORBA. CORB is platform-independent and can be used as a local object. But CORB is very complex and it needs powerful clients and additional network bandwidth.

3.2Network Bandwidth and Protocol

In DSR architecture the connection between client and server can be classified as two types according to different application environments:

- Circuit-switching connection. This type is a modem-to-modem pool connection via PSTN. Circuit-switching connection has stable transmission rate and small time-delay, which can be used in PC, PDA, Wire and Wireless Telephone. For Wire Telephone users additional hardware is needed to realize four function modules of Fig 3.
- Packet-switching connection. Packet-switching connection provides network connection in PC, PDA and Wireless Telephone via Internet so we put our discussion emphasis on this connection type. Internet connection has variant transmission rate and large time-delay compared with PSTN. For the Wireless Telephone case low-speed data channel is used to connect to Internet.

In packet-switching connection, the bandwidth requirement for speech features transmission can be reduced as low as 2.0kbps. But this reduction does sacrifice recognition rate. To achieve higher accuracy, the capability of network bandwidth should be enhanced in both physical layer and logic layer. As for the physical layer, high-speed technologies are preferred, which include ATM, SONET, BROAD IP as backbone and ADSL, HFC, FTTC and FTTH as local connection [11]. In logic layer, a DSR specific protocol has to be defined to guarantee the fast and reliable transmission. DSR data transmission should be based on TCP protocol because it supports connection-oriented, reliable data transmission stream. For DSR's high requirement on data transmission, two issues in DSR network protocol should be taken into account QoS and Congestion Control.

QoS (Quality of Service) describes the requirement of connection so that the traffic can receive best-effort support. DSR connection should be low-delay, low-error-ratio and high-throughput. So at least four QoS parameters should be defined for DSR: table 1. When client requests a connection, the protocol negotiates with the low-level TCP/IP network layer based on these parameters. The latter meets the requirement via route selection algorithms and new technologies such as RSVP, MPLS, or RTP, and etc. When connection is established, client can achieves a bandwidth-optimized route to server.

Throughput(feature data frames rate)
Residual error ratio
Transit delay (line speed)
Priority(urgent or not)

Table 1. Subset of DSR QoS Parameters

The second issue is congestion control. When speech-enabled client is activated, floods of feature frames will be generated. The crowded WAN and memory bound may cause loss of frames. Currently there are two main classes of congestion control schemes: demand reduction scheme and resource creation scheme [14]. Obviously only resource creation scheme can be considered for the transmission congestion problem of DSR. The proposed way is to establish several data connections between client and server if client's LAN bandwidth is available. These connections transmit data across crowded WAN simultaneously. When several connections work at the same time the capacity of network bandwidth will dramaticall increase. The Data Transmitter and SRP will dynamically decide when to establish a new connection according to the data size in client's buffer.

3.3 Distributed Recognition Server

Most of current speech-enabled web servers run on single machine and only can serve one user at one time [6]. In practice, DSR server should quickly answer multiple users ' requests simultaneously. So the effective DSR server has to be constructed by multiple computers and parallel programs to support multiple requests.

3.3.1Scalability of DSR Sever

When the offered load exceeds the capability of DSR, additional hardware should be added to the system. It is hoped that the system performance can increase linearly with the scale of clients and hardware. Generally there are two hardware architectures used to enhance scalability. One is symmetric multiprocessor (SMP) and the other is clustering. SMP expands system resources by adding extra processors, memory and disks to the large, high-performance machine. Cluster systems can be built from commodity PCs. Two principal software models of clustering are used today [15]:

- *Shared disk model:* software running on any system in the cluster may access any resource of the cluster. distributed Lock Manager (DLM) may be used for synchronization and serialization. This model is relativel easy to implement but DLM may limit system scale.
- *Shared nothing model:* each system within the cluster owns a subset of the cluster's resources. Requests from clients are automatically routed to the system that owns the resource. This model has better scalability but its management is complex.

Clustering is well suited to DSR because DSR workloads are highly parallel and the grain size of each workload corresponds to only a few CPU seconds on a commodity PC. Considering maximum scalability, the shared nothing model should be used, in which every computer contains a copy of acoustic and language models in its RAM to save the communication latency between database and the machines.

3.32 Parallel Program of DSR Sever

DSR server can use some parallel program methods such as multithread, variable-share and COOP (Concurrent Object-Oriented Program) to efficiently use these computers [13].

There are several ways to implement such a distributed server of DSR. One method is to construct DSR server over a distributed operating system like MACH, AMOEBA or DCE [12]. Here the proposed method is CORBA.

In section 3.1, CORBA is not thought as suitable for DS client, but it has many advantages in the design and implementation of DSR sever when high-speed LAN technology such as Gigabit Ethernet or FDD is adopted:

- Language- and platform-independence.
- SRP need 't know the location of recognition engines. When an ORB has been configured between SRP and recognition engines, SRP can use ORB as if it were local object to get recognition service.
- SRP need 't know the implementation detail of recognition engines. When the recognition engine is modified, current SRP still can work.
- ORB can be provided by a third party as a common middle-ware.

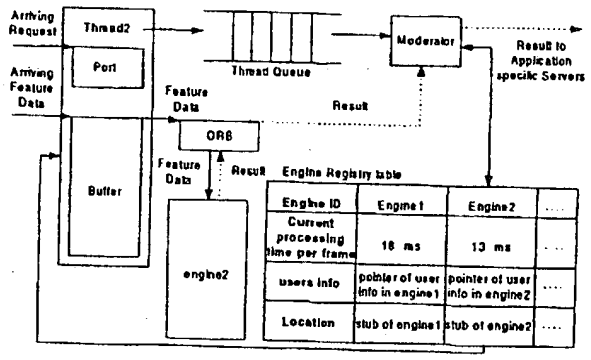

Fig4. Block Diagram of Recognition Management in SRP

The core component in the CORB based implementation of DSR server is SRP. The data/control flow inside SRP is shown in fig 4. In this figure, when a request arrives on SRP, SRP creates a new thread *Thread2* to manage the connection and receive the following feature data. When the data is completely received, the handle of *Thread2* is put into Thread Queue, which is a buffer used to contain all the requesting thread handles. When *Thread2's* handle reaches the top of the queue, this handle is deleted from the queue if there are available engine servers. Otherwise, *Thread2's handle* has to wait.

Moderator decides which engine server is available accordin to a workload schedule scheme. A workload schedule scheme is typically geared to meet certain performance metrics, which include CPU efficiency of engine server, number of jobs (threads), response time of each request (user), amount of processed data, communication throughput and memor requirement, etc. In DSR, the processing time per frame is more practical because it can be used to measure the workload of each engine server efficiently and dynamically. In the other side combining with maximum queue length, the upper limit of the processing time per frame can guarantee the performance of each engine server and acceptable response time of each request.

In fig4. if *engine2* has the minimal value of processing time per frame that is lower than upper limit, so Moderator tells *Thread2* the stub of *engine2* and adds a record in user information table (seen in fig 5). With this stub, *Thead2* sends the feature data to

engine2 via ORB. When *engine2* completes recognition, Moderator receives the result and gets the total processing time of *Thread2* and the number of frames from user information table (seen in fig 5). With these two values, Moderator can calculate the current processing time per frame of *engine2*. Then the processing time per frame in engine registry table is replaced by the current value. T he result and extra user information are forwarded to the application specific server. The latter will understand the result, complete specified processing and then respond to the user.

Pointer to user info table

Thread ID	Thread1	Thread2	.....
Num of frames	300	350	.....
Begin time (us)	12.119.192.274	12.119.192.384	.....
User extra info	service type, source addr..	service type, source addr..	.....

Fig 5. Information Table of User served by engine

4. System Optimization Considerations

DSR System can be optimized to meet different practical requirements. This section proposes some system optimizing suggestions about transmission bandwidth, server workload control and client securit .

4.1Tradeoffs of Recognition Rate and Requirement for Transmission Bandwidth

The network bandwidth is bottleneck of recognition rate in DSR applications. When the speech recognition engine is defined, the detailed experiments should be well done to decide the space quantization of cepstral coefficients based on the best tradeoff point of satisfied recognition rate and reasonable requirement of transmission bandwidth.

4.2 Workload Control of DSR Sever

In DSR server side, system performance should be optimized on several aspects: best utilization of system resources, user-acceptable response time and maximal number of concurrent visitors, etc. Two parameters, Upper Limit of Processing Time Per Frame and Thread Queue Length, should be properly set to meet these requirements. The actual values of the two parameters depend on the system resources and operating system's load control algorithms [16]. System simulation or experiments can be done to get the two values.

4.3Security Consideration in Client

In most cases DSR client has to download functional modules from DSR server. This introduces the problem of security [8]. Several security technologies may be considered which include Authentication protocol, Encryption algorithm and Digital Signature [11]. The proper security policy should depend on tradeoff of client's security requirement and performance.

5. Conclusion

Based on the investigation of ASR systems, networking and distributed computing technologies, a feasible and effective distributed speech recognition system structure is proposed in this paper. This DSR architecture is composed of heterogeneous speech-enabled client devices, distributed recognition server and network transmitting. Realization mechanism is studied thoroughly and system optimization methods are analyzed in order to achieve better system performance. Distributed speech recognition is a challenging work involving many research fields and engineering topics, still much effort must be paid in its grown-up. Our future work will focus on workload balancing and network transmission.

Reference

1. D. Goddeau et al, "Deploying Speech Application over the Web" Proceedings Eurospeech, pp. 685-688, Rhodes, Greece, September 1997.
2. M. Sokolov, "Speaker Verification on the World Wide Web" Proceedings Eurospeech, pp. 847-850, Rhodes, Greece, September 1997.
3. V. Digalakis et al, "Quantization of Cepstral Parameters for Speech Recognition over the World Wide Web", IEEE Journal on Selected Areas in Communications, Vol 17, Num 1, p. 82.
4. The Aurora Project, announced at Telecom 95, "http://gold.ity.int/TELECOM/wt95", Geneva, October 1995.
5. The W3C Voice Browser Working Group, "http://www.w3.mag.keio.ac.jp/Voice/1999/voice-wg-charter.html"
6. L. Julia et al, "http://www.speech.sri.com/demos/atis.html" Proceedings AAAI'97, Stanford, CA, March 1997.
7. D. Stallard, "the BBN SPIN System", presented at the Voice on the Net Conference, Boston, MA, Sep. 1997.
8. S. Bayer, "Embeding Speech in Web Interfaces" Proceedings ICSLP, pp. 1684-1686, Philadelphia, PA, October 1996.
9. Zhemin Tu et al, "Speech Recognition Over the Internet Using Java". *IEEE International Conference on Acoustics, Speech and Signal Processing*, Phoenix, AZ ,1998
10. V. Digalakis et al, "Genones: Optimizing the Degree of Mixture Tying in a Large Vocabulary Hidden Markov Model Based Speech Recognizer" IEEE Trans Speech Audio Processing, pp., July 1996.
11. Andrew S. Tanenbaum "Computer Networks" , PRITENCE HAL
12.Andrew S. Tanenbaum "Distributed Operating System", PRITENCE HA
13. Kai Hwang "Advanced Computer Architecture, Parallelism, Scalability, Programmability (I)", McGraw-Hill, Inc.
14. R. Jain, "Congestion Control in Computer Networks Trends and Issues," IEEE Network, May 1990, pp. 24-30. http://www.cis.ohio-state.edu/~jain/papers/cong_tre.htm
15. Vogels, W. et al. "The Design and Architecture of of the Microsoft Cluster Service – A Practical Approach to High-Availability and Scalabilit ", Proceedings of the 28 th symposium on Fault-Tolerant Computing, Munich, German, June 1998.
16. William Stalling, "Operating Systems, Internals and Design Principles", PRENTICE HAL